UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/712,384	11/12/2003	William John Gallagher	BEAS-01316US3	9603

23910 7590 09/25/2007

FLIESLER MEYER LLP
650 CALIFORNIA STREET
14TH FLOOR
SAN FRANCISCO, CA 94108

EXAMINER	
NGUYEN, PHILLIP H	
ART UNIT	**PAPER NUMBER**
2191	

MAIL DATE	DELIVERY MODE
09/25/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/712,384	GALLAGHER, WILLIAM JOHN	
	Examiner	Art Unit	
	Phillip H. Nguyen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *31 August 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27* is/are pending in the application.

4a) Of the above claim(s) *4-9,14,15 and 19-24* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,10-13,16-18 and 25-27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20070831*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 8/31/2007.

2. Claims 4-9, 14-15 and 19-24have been canceled.

3. Claims 1-3, 12-13, 16-18 and 25 have been amended.

4. Claims 26-27 are newly added

5. Claims 1-3, 10-13, 16-18 and 25-27 remain pending and have been considered below.

Response to Arguments

Specification

1. The amendment filed on 8/31/2007 overcomes the objection to the specification of previous action. Therefore, the objection is withdrawn.

2. The amendment filed 8/31/2007 is objected to under 35 U.S.C. 132(a) because it introduces new matter into the disclosure. 35 U.S.C. 132(a) states that no amendment shall introduce new matter into the disclosure of the invention. The added material which is not supported by the original disclosure is as follows: "any class name and any super class can be selected".

Applicant is required to cancel the new matter in the reply to this Office Action.

Claim Rejections - 35 USC § 112

3. The amendment filed on 8/31/2007 overcomes the rejection to claim 25 of previous action. Therefore, the rejection is withdrawn.

4. Claim 3 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. "any class name and any super class can be selected" is considered new matter because no written description or support in the original specification. For examining purpose, examiner interprets as "class name and super class."

Claim Rejections - 35 USC § 102

5. Applicant's arguments with respect to claims 1-3, 10-13, 16-18 and 25-27 have been considered but are moot in view of the new ground(s) of rejection.

Applicant asserts on page 11 of the amendment that Boehme fails to teach high-level dynamic generation method and does not require generation of adapter source code.

Examiner respectfully disagrees with all the allegations as argued. First, Boehme teaches, "*The detailed description is provided in terms of an implementation in*

the Java programming language (i.e. high level programming language). However, the invention can also be practiced using other object oriented programming languages (i.e. high level programming language)." (See col. 3, lines 47-49). This indicates that Boehme's invention can be practiced using high level dynamic generation method. Second, Boehme teaches, "*It is therefore an object of the present invention to dynamically generate and load adapter classes, without human intervention...*" (See col. 2, lines 61-64). Furthermore, Boehme teaches, "*The invention does not require generation of adapter source code by either the IDE or the programmer. Nor is a compiler required. Executable storage space is also reduced since storing adapter classes in the application executable is not required...Adapter classes and objects are automatically and dynamically generated as required while the application loads and runs.*" (See col. 2, lines 65-67 – col. 3, lines1-4). In other words, Boehme generates source code automatically and dynamically without using IDE or programmer. Col. 4, lines 30-46 shows the source code for adapter class and objects. FIGs. 1-2 further show the process of generating adapter classes and objects.

Applicant further asserts on page 12 of the amendment that Boehme fails to teach a generated class can extended any superclass.

Examiner respectfully disagrees with the allegation as argued. First, examiner believes that the original disclosure dose not disclose "...extended any superclass." Second, examiner does not clearly understand what it means by "any superclass." In other words, examiner does not understand claim 3 as amended. By definition, **Inheritance**: *The transfer of the characteristics of a class in object-oriented*

programming to other classes derived from it. For example, if "vegetable" is a class, the classes "legume" and "root" can be derived from it, and each will inherit the properties of the "vegetable" class: name, growing season, and so on. In other words, they are must be related in order to fulfill the inheritance's purposes. Therefore, claim 3 does not satisfy the purpose of inheritance.

Examiner is entitled to give claim limitations their broadest reasonable interpretation in light of the specification. See MPEP 211 [R-1] Interpretation of Claims-Broadest Reasonable Interpretation. During patent examination, the pending claims must be given their broadest reasonable interpretation consistent with the specification.

Applicant always has the opportunity to amend the claims during the prosecution and broad interpretation by the examiner reduces the possibility that the claims, once issued, will be interpreted more broadly than is justified. In re Prater, 162 USPQ 541, 550-51 (CCPA 1969).

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

7. Claims 1-3, 10, 12, 16-18 and 27 are rejected under 35 U.S.C. 102(e) as being anticipated by Boehme et al. (United States Patent No.: US 6,578,191 B1).

As per claim 1:

Boehme discloses:

- creating a class file container object that stores source code describing a class (see at least col. 3, lines 2-3 "***Adapter classes and objects are automatically and dynamically generated***"; also see col. 4, lines 30-55);
- adding a first source code defining a method to the class stored in the class file container object(see at least col. 4, line 42 "***newClass.addMethod(ACC_PUBLIC, "actionPerformed", "(LactionEvent;)V", bytecodes)";*** also see col. 4, lines 30-55);
- adding a second source code into the method in the class stored in the class file container object (see at least col. 5, lines 49-50 "***code is created for the adapter class initialization methods***"; also see at least col. 5, lines 52-53 "***code is created for the adapter class methods referenced in the adapter class specification***");
- repeating instructions (b) and (c) to populate the class stored in the class file container object (see at least col. 5, lines 49-50 "***code is created for the adapter class initialization methods***"; also see at least col. 5, lines 52-53 "***code is created for the adapter class methods referenced in the adapter***

class..." – In other words, code is created repeatedly for each method in order to fulfill the functionality of the methods);

- generating a tree of statements and expressions based on the class stored in the class file container object (see at least col. 3, line 28 "***Development environments from text editors to IDE's including Visual builders generate source code***..." – IDE interface provides a GUI builder, text or code editor, etc. to generate a tree of statements and expressions);
- using the tree of statements and expressions to generate byte code for the class (see at least col. 4, lines 24-26 "***bytecodes necessary to construct the adapter class, interface, field, methods, and attributes are generated***"); and
- instantiating an instance of the new class file object (see at least col. 4, lines 56-58 "***An instance of the adapter class is instantiated in function block 107, as illustrated by the following source code: Object adapterObject=adapterClass.Newinstance();***").

As per claim 2:

Boehme further discloses:

- wherein creating a class file container object includes:
    - selecting a class name and a super class for a class file (see at least col. 4, line 35 "***newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl***").

As per claim 3:

Boehme further discloses:

- - wherein class name (see at least "***newClass.setSourceFilename("BMLActionEventAdapter")***") and super class can be selected (see at least col. 4, line 35 "***newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl")***").

As per claim 10:

Boehme further discloses:

- wherein the program code (computer code) implements an adaptor class (see at least col. 4, lines 24-25 "***to construct adapter class***"; also see at least col. 4, lines 30-55).

As per claim 12:

Boehme further discloses:

- repeatedly adding a method to the class stored in the class file container object for each method associated with a stub generated for a remote object (col. 4, line 40-43, ***it repeatedly adds methods to the class***).

As per claim 16:

Boehme further discloses:

- wherein the tree of statements and expressions represents at least one method, the at least one method comprising at least one of: a code statement, an expression, a variable and a programming construct (see at least col. 3, lines 28-30 "***Development environments from text editors to IDE's including Visual builders generates source code either automatically or by hand coding***" – IDE provides a window with tree of statements and expressions).

As per claim 17:

Boehme further discloses:

- wherein the tree of statements and expressions forms a known structure or interface when the class file is a known type (see at least col. 3, lines 28-30 "***IDE***").

As per claim 18:

Boehme further discloses:

- wherein the tree of statements and expressions forms a known structure when the class is at least one of an adapter and a proxy type (see at least col. 3, lines 28-30 "***IDE***").

As per claim 27:

Boehme further discloses:

- generating executable code from the byte code by using a class loader (see at least col. 4, lines 48-52 "***the adapter class is then loaded into the running application...Loader ldr=new BMLLoader();***...").

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-3, 10, 12, 16-18 and 27 rejected under 35 U.S.C. 103(a) as being unpatentable over Boehme et al. (United States Patent No.: US 6,578,191 B1), in view of Mellender et al. (United States Patent No.: 4,989,132).

As per claim 1:

Boehme discloses:

- creating a class file container object that stores source code describing a class (see at least col. 3, lines 2-3 "***Adapter classes and objects are automatically and dynamically generated***"; also see col. 4, lines 30-55);
- adding a first source code defining a method to the class stored in the class file container object(see at least col. 4, line 42 "***newClass.addMethod(ACC_PUBLIC, "actionPerformed", "(LactionEvent;)V", bytecodes)";*** also see col. 4, lines 30-55);

- adding a second source code into the method in the class stored in the class file container object (see at least col. 5, lines 49-50 "***code is created for the adapter class initialization methods***"; also see at least col. 5, lines 52-53 "***code is created for the adapter class methods referenced in the adapter class specification***");
- repeating instructions (b) and (c) to populate the class stored in the class file container object (see at least col. 5, lines 49-50 "***code is created for the adapter class initialization methods***"; also see at least col. 5, lines 52-53 "***code is created for the adapter class methods referenced in the adapter class...***" – In other words, code is created repeatedly for each method in order to fulfill the functionality of the methods);
- using the tree of statements and expressions to generate byte code for the class (see at least col. 4, lines 24-26 "***bytecodes necessary to construct the adapter class, interface, field, methods, and attributes are generated***"); and
- instantiating an instance of the new class file object (see at least col. 4, lines 56-58 "***An instance of the adapter class is instantiated in function block 107, as illustrated by the following source code: Object adapterObject=adapterClass.Newinstance();***").

Assuming that Boehme does not explicitly disclose:

- generating a tree of statements and expressions based on the class stored in the class file container object.

However, Mellender discloses:

- generating a tree of statements and expressions based on the class stored in the class file container object (see at least col. 6 – col. 7 "***TABLE 3.1***"; also see col. 7, lines 9-10 "***a successful parse generates a parse tree of the statements of each method in the class***").

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize that generating a tree of statement and expressions is well known in the relevant art and modify Boehme's approach to generate a tree of statement and expressions. One would have been motivated to generate an internal parse tree representation of the class description and its methods, which can be analyzed using a relatively simple set of mutually recursive tree walking routines. In addition, the grammar of the input file is checked and errors are reported to the user (see Mellender col. 5, lines 24-30).

As per claim 2:

Boehme further discloses:

- wherein creating a class file container object includes:
    - selecting a class name and a super class for a class file (see at least col. 4, line 35 "***newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl***").

As per claim 3:

Boehme further discloses:

- - wherein class name (see at least "***newClass.setSourceFilename("BMLActionEventAdapter")***") and super class can be selected (see at least col. 4, line 35 "***newClass.setSuperClassName("com/ibm/bml/EventAdapterImpl")***").

As per claim 10:

Boehme further discloses:

- wherein the program code (computer code) implements an adaptor class (see at least col. 4, lines 24-25 "***to construct adapter class***"; also see at least col. 4, lines 30-55).

As per claim 12:

Boehme further discloses:

- repeatedly adding a method to the class stored in the class file container object for each method associated with a stub generated for a remote object (col. 4, line 40-43, ***it repeatedly adds methods to the class***).

As per claim 16:

Mellender further discloses:

- wherein the tree of statements and expressions represents at least one method, the at least one method comprising at least one of: a code statement, an expression, a variable and a programming construct (see at least col. 6 "***TABLE***

3.1"; see at least col. 6 – col. 7 "***TABLE 3.1***"; also see col. 7, lines 9-10 "***a successful parse generates a parse tree of the statements of each method in the class***").

As per claim 17:

Mellender further discloses:

- wherein the tree of statements and expressions forms a known structure or interface when the class file is a known type (see at least col. 6 "***TABLE 3.1***").

As per claim 18:

Boehme in combination with Mellender disclose all the limitations of claim 17 and

- furthermore, disclose the tree of statements and expressions forms a known structure when the class is at least one of an adapter and a proxy type.

As per claim 27:

Boehme further discloses:

- generating executable code from the byte code by using a class loader (see at least col. 4, lines 48-52 "***the adapter class is then loaded into the running application...Loader ldr=new BMLLoader();***...").

1. Claims 11 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boehme et al. (United States Patent No.: US 6,578,191 B1), in view of Cohen et al. (United States Patent No.: 6,011,918).

As per claim 11:

Boehme does not explicitly disclose:

- wherein the program code implements a proxy class.

However, Cohen discloses an analogous computer program product wherein the program code implements a proxy class ("**proxy class is called X, ... will execute on the local machine**" Col 15, line 32-34). Therefore, it would have been obvious to one having an ordinary skill in the art to modify Boehme's approach to implement proxy class. One having an ordinary skill in the art would have been motivated to implement proxy class so that **each of the constructed methods make corresponding remote method calls using Java's RMI function** (see Cohen Col 14, line 5-10).

As per claim 13:

Boehme does not explicitly disclose

- wherein the program code for repeatedly adding a method to the class file object for each method associated with a stub generated for a remote object includes program code for: determining a number of methods associated with the stub in a remote interface.

However, Cohen discloses an analogous computer program product for determining a number of calling methods ("**the weighting of the relationships between the programmed methods is performed by determining for each calling programmed method**" Col 3, line 52-55). Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Boehme's approach to determine the number of method added to the class file object. One having an ordinary skill in the art would have been motivated **to determine the number of method by include a counter or an indicator that increment each time a method is adding to the class file object so it would know when it reaches the end of the adding iteration process**.

10. Claims 25 and 26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boehme et al. (United States Patent No.: US 6,578,191 B1), in view of Mellender et al. (United States Patent No.: 4,989,132).

As per claim 25:

Boehme does not explicitly disclose:

- wherein the dynamically generated code is used for remote method invocation skeleton, remote method invocation stubs, wrappers for database connections and proxies used to enforce call-by-value semantics.

However, it would have obvious to one having an ordinary skill in the art at the time the invention was made to recognize that remote method invocation skeleton, remote

method invocation stubs, wrappers for database connections and proxies used to enforce call-by-value semantics are well known in the relevant art.

Therefore, one would have been motivated to use these well known technique to communicate remotely with other Java objects and to intercepting calls between objects to improve convenience, compatibility or security.

As per claim 26:

Assuming Boehme does not explicitly discloses:

- wherein dynamically generated code exists for the life of a server it resides upon.

However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize that the generated adapter class exists until it fulfills its purpose.

Therefore, one would have been motivated to keep the generated adapter class for the life of a server to fulfill the dynamic code generation purposes.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phillip H. Nguyen whose telephone number is (571) 270-1070. The examiner can normally be reached on Monday - Thursday 10:00 AM - 3:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

PN
9/12/2007

WEI ZHEN
SUPERVISORY PATENT EXAMINER